REMARKS

7/16/03

Claims 1, 3-12 and 14-48 are pending in the present application. Reconsideration of the claims is respectfully requested in view of the following remarks.

I. 35 U.S.C. § 103, Alleged Obviousness Based on Ng and Sarkar

The Office Action rejects claims 1, 3-4, 7-12, 14-15, 17-19, 25-26, 31-34, 39-42, 44 and 46-48 under 35 U.S.C. § 103(a) as being unpatentable over Ng et a. (U.S. Patent No. 6,385,618) in view of Sarkar (U.S. Patent No. 6,418,448).

As to independent claim 1, the Office Action states:

> With respect to claim 1, Ng discloses determining a structure of the relational database (database schema of a relational database: col. 4, lines 23-27 and lines 35-36), wherein determining the structure of the relational database includes invoking a database meta-information class object associated with the relational database (database metadata is called by the tool via JDBC: col. 7, lines 60-67 and col. 8 lines 1-18; also see fig. 9); determining a delete action based on the structure of the relational database and generating database modification commands based on the determined delete action and sending the database modification commands (abstract, col. 2, lines 36-49; col. 4, lines 22-54; also see col. 3, lines 35-38 and lines 45-59).

Office Action dated April 22, 2003, page 3.

Claim 1, recites:

> 1. A method of deleting object data from a relational database, comprising:
>
> determining a structure of the relational database, wherein determining the structure of the relational database includes invoking a database meta-information class object associated with the relational database;
>
> determining a delete action based on the structure of the relational database;
>
> generating database modification commands based on the determined delete action; and
>
> sending the database modification commands to a relational database server, wherein the relational database server deletes the object data from the relational database based on the database modification commands. (emphasis added)

Ng is directed to an object relational mapping tool for integrating both modifications to an object model and modifications to a database into source code. Ng teaches that the object relational mapping tool maps a database by first querying the database to determine its schema and then creating an internal data structure representing that schema. From this data structure, the object relational mapping tool creates an object model containing all of the information necessary to generate classes and then creates source code containing a number of Java classes that may be used by a programmer to interface with the database (column 4, lines 24-31).

At some time during the lifetime of the object model, the programmer may add customizations to the object model that will be reflected in the source code. Additionally, the database administrator may make changes to the database schema. After the database schema is changed, the programmer may wish to update their source code to reflect the schema change while maintaining their customizations. To accomplish this, the object relational mapping tool of Ng creates a new database data structure, compares it to the original database data structure, and notes any differences. These changes are then incorporated into the existing object model and new source code is generated from this modified object model. As a result, both the programmer's customizations to the object model as well as the changes to the schema made by the database administrator are integrated into the new source code (column 4, lines 32-54).

Thus, Ng teaches a system for updating an original object model, possibly having customizations from a programmer, with only the changes to the database made by a database administrator, as represented in the new database data structure. The result of the update of the original object model is a combination of the original object model, any customizations added by the programmer, and the changes to the database made by the administrator. Once the original object model is updated, new source code is generated.

Nowhere in Ng is it taught to determine a delete action based on the structure of a relational database. While Ng teaches determining the structure of a database and generating a database data structure based on the determined structure of the database, Ng never mentions that a delete action is determined based on the determined structure of the database. To the contrary, Ng teaches that an object model is generated based on the database data structure.

The Office Action alleges that this feature of claim 1 is taught by Ng in the abstract, at column 2, lines 36-49, column 4, lines 22-54, and column 3,lines 35-38 and 45-59. The abstract of Ng reads as follows:

> In accordance with methods and systems consistent with the present invention, an improved object-relational mapping tool is provided that generates source code containing classes which preserve both changes to the database schema as well as customizations to a preexisting version of the classes. This functionality alleviates the programmer from having to recreate their changes to the classes when the database changes, thus saving significant development time over conventional systems.

As is clear from the above, there is no mention of determining a delete action based on the structure of a relational database provided in the Abstract of Ng. While the abstract mentions that the object relational mapping tool preserves "both changes to the database schema as well as customizations to preexisting version of the classes" this does not teach or suggest determining a delete action based on a determined structure of a relational database. Taking this statement in context with the rest of the description of Ng, it is clear that what is stated here is the mechanisms previously discussed above for updating an object model such that the result of the update is a combination of an original object model, customizations made by the programmer, and changes to the database structure made by an administrator. Nowhere is a delete action ever mentioned in Ng, let alone determining a delete action based on a determined structure of a relational database.

None of the other cited sections of Ng provide any teachings regarding this feature either. Column 2, lines 36-49 reads as follows:

> Although beneficial to programmers, conventional object-relational mapping tools suffer from a limitation. When a programmer runs the object-relational mapping tool, it generates source code with classes that reflect the structure of the database at that time. However, during the lifetime of the database, it is common for a database administrator to change the schema of the database (e.g., add a new field or table). Likewise, it is common for the programmer to update the classes in the source code (e.g., change a field name or delete a field). As such, both the classes and the database tend to evolve and change over time. Conventioanl object-relational mapping tools, however, are of little help in such situations. These tools can only remap the database to

generate classes that contain the database modifications, but which do not contain the programmer's modifications.

This section merely teaches that conventional object-relational mapping tools suffer from only being able to remap the database to generate classes that contain the modifications to the database and do not include the programmer's modifications to the source code. While the word "delete" is in this section, the deletion referred to is the deletion, performed by a programmer, of a field from source code. There is no teaching or suggestion in this portion of Ng regarding determining a delete action based on the structure of a relational database.

Column 4, lines 22-54 read as follows:

> In accordance with methods and systems consistent with the present invention, the improved object-relational mapping tool maps a database by first querying the database to determine it schema and then be creating an internal structure (known as the "database data structure") representing that schema. From this data structure, the object-relational mapping tool creates an object model containing all of the information necessary to generate classes and then creates source code containing a number of Java classes that may be sued bya programmer to interface with the database.
>
> At some point during the lifetime of the object model, the programmer may add customizations to the object model (e.g., rename a field) that will be reflected in the source code, and the database administrator may likewise make a change to the database schema (e.g., add a column). After the database schema has been changed, the programmer may wish to update their source code to reflect the schema change while maintaining their customizations. To accomplish this goal, the object-relational mapping tool, in accordance with methods and systems consistent with the present invention, imports the new schema, creates a new database data structure, and compares the original (or preexisting) database data structure with this newly created one, noting any differences. Having noted the differences, the object-relational mapping tool has isolated the changes to the schema, and it then incorporates these changes into the existing object model and generates new source code. As a result, both the programmer's customizations to the object model (reflected in the old version of the source code) as well as the changes to the schema made by the database administrator are integrated into the new source code, thus saving the programmer significant development time over conventional systems.

There are no teachings or suggestions in this section of Ng regarding determining a delete action based on a determined structure of a relational database. While this section of Ng states that a database schema may be modified and that these modifications will be identified and represented in a new object model and new source code, there is nothing in this, or any other, section of Ng regarding determining a delete action based on a determined structure of a relational database.

Column 3, lines 35-38 and 45-59 read as follows:

> ...an improved object-relational mapping tool is provided that generates source code containing classes which preserve both changes to the database schema as well as customizations to a preexisting version of the classes.

(Column 3, lines 35-38)

> ...a first data structure reflecting a logical structural relationship among data in a data source. This method creates a second data structure reflecting a modified version of the logical structural relationship among the data in the data source and compares the first data structure with the second data structure to isolate the modifications made to the logical structural relationship among the data so that source code can be generated to reflect the modifications.
>
> In accordance with methods consistent with the present invention, a method is provided in a computer system having an object model reflecting a logical structure of a data source. This method receives customizations into the object model, receives an indication that the data source has been modified, and incorporates the modifications into the object model while preserving the customizations.

(Column 3, lines 45-59).

As with the other cited portions of Ng, these sections also do not teach or suggest determining a delete action based on a determined structure of a relational database. Lines 35-38 of column 3 merely teaches preservation of changes to a database schema as well as customizations to a preexisting version of classes. These features of Ng have been discussed above. Lines 45-59 of column 3 merely teach the comparison of the database schema data structures to isolate the modifications made to the structure of the database and the incorporation of these isolated modifications into the object model while

keeping the customizations made to the object model. Nothing in these sections mentions or even suggests to determine a delete action based on a determined structure of a relational database.

Furthermore, since Ng does not teach or suggest determining a delete action based on a determined structure of a relational database, Ng cannot be found to teach or suggest the feature of generating database modification commands based on a determined delete action or sending the database modification commands to a relational database server, wherein the relational database server deletes object data from the relational database based on the database modification commands.

The Office Action alleges that the generating of database modification commands based on a determined delete action is taught at the same sections as determining a delete action based on the structure of the relational database, i.e. abstract, at column 2, lines 36-49, column 4, lines 22-54, and column 3,lines 35-38 and 45-59. These sections have been addressed above and it is clear that none of these sections teach determining a delete action based on a determined structure of a relational database or generating database modification commands based on a determined delete action. While Ng mentions modifications to a database schema, these modifications are not modification commands that are generated based on a delete action that is determined based on a determined structure of a relational database. The modifications in Ng are changes made by an administrator to the database schema that must be reflected in the object model. They are not modification commands that are generated based on a determined delete action.

With regard to the feature of sending the database modification commands to a relational database server, the Office Action admits that Ng does not teach this feature. In fact, Ng does not teach this feature because Ng is not directed to a system for deleting object data from a relational database based on database modification commands generated based on a delete action determined based on a determined structure of a relational database. To the contrary, Ng is directed to a system for ensuring that the customizations to an object model made by a programmer, and the modifications to the database schema made by a database administrator, are reflected in an updated version of the object model which is used to update source code.

The Office Action alleges that this feature of claim 1 is taught by Sarkar. However, Sarkar does not provide for the deficiencies of Ng. That is, Sarkar does not teach determining a delete action based on a determined structure of a relational database or generating database modification commands based on a determined delete action. Sarkar teaches a method and apparatus for processing markup language specifications for data and metadata used inside multiple related internet documents to navigate query and manipulate information from a plurality of object relational database over the world wide web.

While Sarkar may teach loading Java classes in a database server as alleged by the Office Action, there is no teaching or suggestion in Sarkar regarding determining a delete action based on the structure of a relational database, generating database modification commands based on the determined delete action, or sending the database modification commands to the relational database server. Loading of classes in a database server does not teach or suggest sending database modification commands that are generated based on a delete action determined based on a determined structure of a relational database.

Furthermore Sarkar is directed to solving a completely different problem than that of Ng. Ng. is directed to preserving object model customizations while accommodating changes to database schema. Sarkar is directed to a system for navigating, querying and manipulating information from a plurality of object relational databases over the Internet. One of ordinary skill in the art would not have been motivated to combine these two systems at least because they are directed to two disparate fields of technology.

Moreover, neither reference teaches or suggests the desirability of incorporating the subject matter of the other reference. That is, there is not motivation offered in either reference for the alleged combination. The Office Action alleges that the motivation for the combination is to "obtain database server of a object relational database locating of elements inside component relational schema with Java classes." While Sarkar mentions locators of elements inside component relational schema, there is no teaching or suggestion that this is a reason why the system of Sarkar should be combined with the system of Ng. Moreover, there is no suggestion in Ng of the desirability to be able to locate "elements inside component relational schema with Java classes." To the contrary, the only teaching or suggestion to even attempt the alleged combination is based on a

prior knowledge of Applicants' claimed invention thereby constituting impermissible hindsight reconstruction using Applicants' own disclosure as a guide.

Furthermore even if one were somehow motivated to combine Sarkar with Ng, the result would not be the invention as recited in independent claim 1. Since neither reference alone teaches or suggests determining a delete action based on a determined structure of a relational database, generating database modification commands based on a determined delete action, or sending the generated database modification commands to a relational database server, the alleged combination would still not teach or suggest these features.

Independent claims 12 and 46 recite similar features to those of claim 1 and thus, are allowable over the alleged combination of Ng and Sarkar for similar reasons. That is, claim 12 recites that the data processor determines a delete action based on the structure of the relational database, generates database modification commands based on the determined delete action and sends the database modification commands to the relational database storage device. Claim 46 recites determining one or more default delete actions based on the structure of the relational database. Claim 46 recites additional features that are not addressed in this rejection.[1]

In view of the above, Applicants respectfully submit that neither Ng nor Sarkar, either alone or in combination, teach or suggest the features of independent claims 1, 12 and 46. At least by virtue of their dependency on claims 1, 12 and 46, respectively, neither Ng nor Sarkar, either alone or in combination, teach or suggest the features of dependent claims 3-4, 7-11, 14-15, 17-19 and 47-48. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 1, 3-4, 7-12, 14-15, 17-19 and 46-48 under 35 U.S.C. § 103(a).

In addition to the above, neither Ng nor Sarkar teach or suggest the specific features recited in dependent claims 3-4, 7-11, 14-15, 17-19, 25-26, 31-34, 39-42, 44 and 47-48. For example, with regard to claims 3 and 14, neither reference teaches or suggests a meta-information class object that encapsulates a dependency structure of the relational database. The Office Action alleges that this feature is taught at column 6, lines 12-20 of

[1] While the statement of this rejection includes claim 46, there is no actual rejection of claim 46 under this section of the Office Action. To the contrary, claim 46 is rejected under 103(a) based on Ng alone (see page 12 of the Office Action).

Sarkar, however this section of Sarkar merely teaches the encapsulation of business logic in Java classes, not the dependency structure of a relational database.

Regarding claims 4 and 15, neither reference teaches a meta-information class object that includes a delete action identifier for each dependent table of a plurality of tables in the relational database. The Office Action alleges that this feature is taught by Ng at column 3, lines 62-67, column 7, lines 60-67, column 8, lines 1-17 and Figure 9. Applicants respectfully disagree.

Column 3, lines 62-67, column 7, lines 60-67 and column 8, lines 1-17 read as follows:

> The secondary storage device contains a database having a logical structure comprising tables with rows and columns. The memory contains a first database data structure reflecting the logical structure of the database and an object modle containing objects based on the first database data structure.

(Column 3, lines 62-67)

> Figure 9 depicts a flowchart or the states performed when importing the database schema. Below, the object-relational mapping tool utilizes a number of methods which are found on the DatabaseMetaData interface of JDBC. The first state performed by the object-relational mapping tool is to call the GetTable method of the JDBC interface, which returns a description of the tables of the database (state 902). After retrieving this table information, the object-relational mapping tool selects one of the tables (state 904) and invokes the GetColumns method on the JDBC interface, returning a descriptions of all of the columns in that table (state 906). Next, the object-relational mapping tool invokes the GetPrimaryKeys method to receive the primary key for the table (state 908). After obtaining the primary key, the object-relational mapping tool invokes the GetImportedKeys method to obtain information regarding the foreign keys (state 910). After invoking this method, the object-relational mapping tool determines if there are additional tables to be processed (state 912). If so, processing continues to state 904. Otherwise, the object-relational mapping tool constructs a database data structure, like the one shown in FIG. 7, from all of the information received in the previous states (state 914).

(Column 7, line 60 to column 8, line 17)

None of these sections even mention a delete action identifier, let alone that a delete action identifier is included in a meta-information class object for each dependent

table of a plurality of tables in a relational database. The cited sections are simply not relevant to the features recited in claims 4 and 15.

With regard to claims 7 and 17, neither reference teaches or suggests a file describing the structure and delete actions for tables in a relational database or that a database meta-information class is generated based on such a file. The Office Action alleges that this feature is again taught at column 3, lines 62-67, column 7, lines 60-67, column 8, lines 1-17 and Figure 9 of Ng (these sections are reproduced above). However, these sections are equally irrelevant to the features of claim 7 as they are to the features of claim 4. Nowhere in any of the cited sections of Ng is it ever taught or suggested to have a file that describes the structure and delete actions for tables in a relational database or the generation of a database meta-information class based on such a file. While Ng teaches a database data structure that identifies the schema of the database, there is nothing in Ng that teaches that this database data structure includes a description of delete actions for tables in the relational database or that a meta-information class is generated based on this database data structure.

Regarding claim 8, neither reference teaches or suggests that the file describing the structure and delete actions for tables in a relational database is an Extended Markup Language file. While Sarkar teaches Extensible Markup Language, there is no teaching in Sarkar to use Extensible Markup Language to generate a file describing the structure and delete actions for tables in a relational database.

With regard to claims 9-11 and 18-19, neither reference teaches or suggests that the file is further generated based on user input to override default delete action identifiers in the file (claims 9 and 18), the file being generated based on user input to insert one or more delete constraints in the file for one or more of the tables in the relational database (claim 10 and 19), or that the database modification commands are SQL statements (claim 11). The Office Action alleges that these features are taught by Ng at column 3, lines 62-67, column 7, lines 60-67, column 8, lines 1-17, Figure 9, column 7, lines 16-26. With the exception of column 7, lines 16-26, these sections have been reproduced and addressed above with regard to claims 4 and 15. Column 7, lines 16-26 of Ng read as follows:

> Referring again to FIG. 5, the first state performed by the object-relational mapping tool is to import the database schema (state 502). In this state, importation of the database schema is facilitated through the use of the Java™ Tm database connectivity product (JDBC) available from Sun Microsystems of Palo Alto, Calif. JDBC is a Java™ application program interface (API) for executing structured query language (SQL) statements. It consists of a set of classes and interfaces written in the Java™ TM programming language.

None of these sections even mention a file that describes the structure of a relational database and delete actions for tables in the relational structure, let alone overriding default delete action identifiers in such a file based on user input (claims 9 and 18). Moreover, none of these sections teach or suggest delete constraints, let alone user input that inserts one or more delete constraints in a file for one or more tables in a relational database (claims 10 and 19). Further, while Ng may teach SQL statements, there is no teaching or suggestion in Ng to generate SQL statements that are modification commands based on a determined delete action that is determined based on a structure of a relational database.

Regarding claims 25-26, 33-34 and 41-42, as previously discussed above, neither reference teaches or suggests a user input to override default delete action identifiers in a file that describes the structure and delete actions for tables in the relational database (claims 25, 33 and 41) or user input to insert one or more delete constraints in such a file (claims 26, 34 and 42). These features were addressed above with regard to claims 9-11 and 18-19.

With regard to claims 31 and 29, neither reference teaches a file that describes the structure and delete actions of tables in a relational database. This feature was addressed with regard to claim 8 above and claims 31 and 39 distinguish over the alleged combination of references for similar reasons.

Regarding claim 44, the specific features recited in this claim are similar to those of claims 3 and 14 and thus, claim 44 distinguishes over the alleged combination of Ng and Sarkar for similar reasons as noted above.

Regarding claims 47 and 48, neither reference teaches user input for overriding one or more default delete actions or inserting one or more delete action constraints. Similar features have been addressed above with regard to claims 25-26, 33-34 and 41-42

and claims 47 and 48 are allowable over the alleged combination of references for similar reasons as noted above.

Thus, claims 3-4, 7-11, 14-15, 17-19, 25-26, 31-34, 39-42, 44 and 47-48 are allowable over the alleged references both by virtue of their dependency and by virtue of the specific features recited in these dependent claims. Therefore, Applicants respectfully request withdrawal of the rejection of claims 3-4, 7-11, 14-15, 17-19, 25-26, 31-34, 39-42, 44 and 47-48 under 35 U.S.C. § 103(a).

II. 35 U.S.C. § 103, Alleged Obviousness Based on Ng, Sarkar and Crus

The Office Action rejects claims 5-6, 16, 22-23, 29-30, 37-38 and 45 under 35 U.S.C. § 103(a) as being unpatentable over Ng in view of Sarkar and further in view of Crus (U.S. Patent No. 4,947,320). This rejection is respectfully traversed.

With regard to claims 5-6, 16, 22-23, 29-30, 37-38 and 45, the Office Action admits that neither Ng nor Sarkar teach that a delete action or the delete action identifier is one of cascade delete and nullify columns. The Office Action alleges that Crus teaches this feature at column 5, lines 3-67, column 6, lines 1-36, column 16, lines 60-67, column 17, lines 1-67 and column 18, lines 1-18. These sections are reproduced below:

> Overview of Record-Level Constraint Enforcement
>
> According to this invention, referential constraints are enforced at the time each record is manipulated, such as by a load, insert, update or delete command. Whenever a manipulation is invoked, the table descriptor that describes the table to which the record belongs is consulted to see if there are any referential constraints to be enforced for the specific operation being performed. The presence of constraints is indicated by an identifier in the table descriptor of the first relationship descriptor representing a constraint on that table.
>
> If a referential constraint affecting the record is identified, then the referential constraint is enforced. The exact algorithm for checking is different for each type of manipulation (insert/update/delete), with each algorithm being described in detail below. If any referential constraint associated with the record's manipulation is violated, then all manipulations done during the statement or transaction up to the detection of the violation are undone or "backed out", and an error return code

indicating the specific constraint violation is passed back to the user or application.

FIG. 6 provides a schematic overview of the preferred embodiments of this invention's method of record-level constraint enforcement for the three major SQL commands which manipulate relational data base tables–INSERT, UPDATE and DELETE.

Constraint Enforcement During Insert Operations

As shown by reference numeral 28 of FIG. 6, when a table is manipulated by inserting new records, each record is inserted and its constraints enforced separately from the other records in the insert operation. First the record is inserted 30 into the table, and all indexes for that table are connected to the new record, i.e., an entry for the new record is inserted in each index. Then the...

(Column 5, lines 1-37)[2]

...referential constraint check is performed 32 for each constraint in which the record is a dependent, to ensure that the parent table contains a record with a primary key value matching the foreign key value contained in the newly inserted record. This is done by searching the primary index of the parent table, which is identified in the relationship descriptor, for a key value that matches the foreign key of the row being inserted. If a matching index key is found, it means that a row exists in the parent table with a primary key that matches the foreign key, and the referential constraint is satisfied. This check is repeated for each additional constraint in which the record being inserted is a dependent.

If a matching primary key value is not found 34 for an inserted record subject to a referential constraint, then that constraint is violated and error processing 36 is performed. Error processing consists of backing out from all manipulated tables and indexes those manipulations that occurred on behalf of the insert command up to the point of error, and returning an error return code to the user. Methods of backing out aborted commands and operations are well known in the prior art.

If all records are inserted without constraint violations, the operation is successful 37, and the changes to the data base are complete.

The method of inserting first and checking second allows this invention to handle self-referencing rows normally, without requiring special programming. By inserting the primary key and associated index entry first, the primary key that matches the new foreign key in the same row will always be found when the constraint checking is performed after the insert, thus satisfying the constraint.

[2] It should be noted that column 5 of the Crus reference ends at line 37, not at line 67 as stated in the Office Action.

Table 1 shows a pseudocode implementation of this invention's method for enforcing referential constraints during inserts.

(Column 6, lines 1-36)

Constraint Enforcement During Delete Operations

In enforcing referential constraints during delete operations, the problem is that primary key values may be deleted, which requires that any dependent rows with matching foreign keys be checked for violations of referential integrity. The method used to find foreign keys in each dependent table matching the deleted primary key in the parent table is the same as described...

(Column 16, lines 60-67)

above under the heading "Constraint Enforcement on Updates of Primary Keys". The index, if any, on the foreign key is searched, otherwise the dependent table is scanned. When a foreign key is found the action taken depends on the particular delete rule specified when the constraint was created.

There are three delete rules: DELETE RESTRICT, DELETE SET NULL, and DELETE CASCADE. The DELETE RESTRICT rule prevents the deletion of a primary key value if any matching foreign key value exists, and is enforced in the same manner as updates to primary keys. If a matching foreign key value is found then a constraint violation has taken place and the operation is aborted and backed out. If no matching foreign key value is found, the constraint on that record is upheld and the record delete is successful.

The DELETE SET NULL rule sets all nullable columns of the foreign key to null value upon deletion of the primary key value matching the old foreign key value. DELETE SET NULL changes all formerly matching foreign key values in the dependent table to null.

DELETE CASCADE operates similarly to DELETE SET NULL, but instead of setting the foreign key values to null, DELETE CASCADE deletes all records in the dependent table containing the formerly matching foreign key values. These deletions are then "cascaded" from the dependent table to its dependent tables (those tables which have the deleted record's table as their parent table), with the resulting deletes of...

(Column 17, lines 1-30)[3]

...dependent records being known as "cascade deletes." The table ("record-type") descriptor of the deleted record's dependent table is

[3] It should be noted that column 17 of the Cruz reference ends at line 30, not at line 67 as stated in the Office Action.

> consulted to see if it also a parent table in another referential constraint. If yes, a new delete constraint enforcement operation is initiated in which the the cascade-deleted record is treated as a parent, and its delete rule (RESTRICT, SET NULL, or CASCADE) is enforced. If its delete rule is DELETE CASCADE, the same process just described is repeated. Otherwise, a DELETE SET NULL or DELETE RESTRICT operation is performed. DELETE CASCADE is thus a recursive operation that propagates to all dependent tables connected to a given parent table from which a record is being deleted. If a DELETE RESTRICT error is encountered along the way, then the entire delete operation, including all of its cascaded deletes, is backed out.

(Column 18, lines 1-18)

While Crus teaches a delete set null and a delete cascade operation in column 17, there is no teaching or suggestion to identify a delete set null or a delete cascade operation based on a determined structure of a relational database, as determined from a meta-information class object associated with a relational database. Moreover, there is not teaching or suggestion in Crus to include a delete set null or a delete cascade operation identifier, for each dependent table of a plurality of tables in a relational database, in a meta-information class object. Therefore, there is no teaching or suggestion in Crus regarding a delete action (claims 6, 22-23, 29-30, 37-38 and 45) or delete action identifier (claims 5 and 16) being one of a cascade delete and nullify columns delete as recited in the above claims.

Thus, the alleged combination of Ng, Sarkar and Crus does not teach or suggest all of the features of claims 5, 6, 16, 22-23, 29-30, 37-38 and 45. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 5, 6, 16, 22-23, 29-30, 37-38 and 45 under 35 U.S.C. § 103(a).

III. 35 U.S.C. § 103, Alleged Obviousness Based on Ng

The Office Action rejects claims 20-21, 24, 27-28, 35-36, 43 and 46 under 35 U.S.C. § 103(a) as unpatentable over Ng.

With regard to claim 20, the Office Action states that Ng teaches the features of this claim at the same portions of Ng previously discussed above. Claim 20 recites:

20. A method of generating a class for deletion of data representations of objects in a relational database, comprising:

determining a structure of the relational database;

determining one or more delete actions based on the structure of the relational database; and

generating the class object based on the determined structure and the determined one or more delete actions.

It should first be noted that the rejection of claim 20 is not based on obviousness despite the statement of the rejection. The text of the rejection provides no indication as to what the Examiner believes is not taught in Ng, how Ng is being modified, the basis for such modification, or the motivation for such a modification. Therefore, the Office Action has not established its burden of presenting a prima facie case of obviousness with regard to claim 20.

Furthermore, as previously noted above with regard to claim 1, Ng does not teach or suggest determining a delete action based on a determined structure of a relational database. Moreover, Ng does not teach or suggest generating a class object based on a determined structure and one or more determined delete actions. While Ng teaches that an object model and source code may be generated from the database data structure that represents the database schema, there is no teaching of suggestion in Ng to generate the object model or the source code based on one or more determined delete actions. None of the cited sections of Ng even mention the possibility of such a feature, let alone explicitly teach or suggest such a feature.

Claims 27 and 35 recite similar features and thus, distinguishes over Ng for similar reasons as noted above.

Regarding independent claim 43, Ng does not teach or suggest a meta-information class for determining a structure of the relational database and one or more delete actions based on the structure of the relational database or a database meta-information generator class for generating the class object based on the determined structure and the determined one or more delete actions. Similar features are noted above with regard to claims 20, 27 and 35 and claim 43 distinguishes over Ng for similar reasons.

With regard to independent claim 46, Ng does not teach or suggest determining one or more default delete actions based on the structure of the relational database,

receiving user input to modify the one or more default delete actions, or generating a class object based on the determined structure, the determined one or more delete actions and the user input.

Furthermore, there is no suggestion in Ng to make the necessary modifications to arrive at the invention recited in independent claims 20, 27, 35, 43 and 46. Moreover, the Office Action fails to provide any motivation for modifying the Ng reference. Rather than offering a reason why it would be obvious to modify the teachings of Ng is a specific manner, the Office Action merely states that "it would have been obvious to a person of ordinary skill in the art at the time the invention was made to utilize the teachings of Ng for changing the structure of relational database by database administrator and integrating to the database system (col. 4, lines 45-54) in the deletion of object from an object relational system in a customizable and database independent manner environment." Where is the reason why it would be obvious to do this? Furthermore, this statement does not seem to have anything to do with the present claims but is some random statement removed from the Ng reference. There simply is no motivation to modify Ng in any way to arrive at Applicants' claimed invention.

In view of the above, Applicants respectfully submit that Ng does not teach or suggest all of the features of independent claims 20, 27, 35, 43 and 46. At least by virtue of their dependency on claims 20, 27 and 35, respectively, Ng does not teach or suggest the features of dependent claims 21, 24, 28 and 36. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 20-21, 24, 27-28, 35-36, 43 and 46 under 35 U.S.C. § 103(a).

In addition to the above, Ng does not teach or suggest the specific feature recited in dependent claims 21, 24, 28 and 36. For example, Ng does not teach encapsulating the structure of a relational database and one or more delete actions in a class object, as recited in claims 21, 28 and 36. Ng also does not teach or suggest one or more delete actions being determined from a file describing the structure and delete actions for table sin the relational database, as recited in claim 24. Similar features to these have been discussed above with regard to the alleged combination of Ng, Sarkar and Crus and the fact that Ng does not teach these features has been addressed above. Therefore,

Unofficial

7/16/03

Applicants direct the Examiner's attention to the previous arguments for treatment of these features.

IV. Conclusion

It is respectfully urged that the subject application is patentable over Ng, Sarkar and Crus and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

DATE: July 16, 2003

Stephen J. Walder, Jr.
Reg. No. 41,534
Carstens, Yee & Cahoon, LLP
P.O. Box 802334
Dallas, TX 75380
(972) 367-2001
Attorney for Applicant